Multi-streaming and multi-GPU optimization for a matched pair of Projector and Backprojector

Nicolas Georgin, Camille Chapdelaine, Nicolas Gac, Ali Mohammad-Djafari, Estelle Parra-Denis

Multi-streaming and multi-GPU optimization for a matched pair of Projector and Backprojector

Nicolas Georgin[a], **Camille Chapdelaine**[a,b], **Nicolas Gac**[b], **Ali Mohammad-Djafari**[c], **Estelle Parra**[a]

[a]Safran Tech, Signal and Information Technologies Department, Rue des Jeunes Bois, Châteaufort, 78114 Magny-Les-Hameaux, France

[b]Laboratoire des signaux et systèmes, CNRS, CentraleSupélec-Université Paris-Sud, Gif-sur-Yvette, France

[c]International Science Consulting & Training, 91440 Bures-sur-Yvette, France

Abstract. Iterative reconstruction methods are used in X-ray Computed Tomography in order to improve the quality of reconstruction compared to filtered backprojection methods. However, these methods are computationally expensive due to repeated projection and backprojection operations. Among the possible pairs of projector and backprojector, the Separable Footprint (SF) pair has the advantage to be matched in order to ensure the convergence of the reconstruction algorithm. Nevertheless, this pair implies more computations compared to unmatched pairs commonly used in order to reduce the computation time. In order to speed up this pair, the projector and the backprojector can be parallelized on GPU. Following one of our previous work, in this paper, we propose a new implementation which takes benefits from the factorized calculations of the SF pair in order to increase the number of data handled by each thread. We also describe the adaptation of this implementation for multi-streaming computations. The method is tested on large volumes of size 1024^3 and 2048^3 voxels.

Keywords: Computed Tomography, Separable Footprint (SF), Graphical Computed Unit (GPU), CUDA.

*Nicolas Georgin, nicolas.georgin@supelec.fr *Camille Chapdelaine, camille.chapdelaine@safrangroup.com

1 Introduction

Iterative reconstruction methods can greatly improve the quality of reconstruction in cone-beam Computed Tomography (CBCT), particularly for low-dose acquisition.[1] Unfortunately, these methods suffer from a very long computation time which is conditioned by the speed of projection and backprojection operators. Theoretically, the backprojector is the adjoint operator of the projector. Nevertheless, unmatched operators are very often used in order to alleviate the computational burden, especially for 3D reconstruction. This corresponds to an approximation which can hinder the convergence of the reconstruction algorithm.[2] In order to ensure the convergence, computationally-efficient pairs have been proposed.[1,3] Among these pairs, the Separable Footprint (SF) pair[1] approximates the footprint of a voxel onto the detector as a separable function with respect to the transaxial and the axial directions. Thanks to this approximation, the computations can be factorized so SF projection and backprojection can be computed in a reasonable time. Nevertheless, this matched pair requires more computations than unmatched pairs, even parallelized on Graphical Processing Unit (GPU).[4–6]

This paper describes how, based on our previous work,[6] the SF projection and backprojection operators are accelerated using multi-streaming and multi-GPU computation. We propose a new design of the projector and backprojector kernels, in which each thread computes the projections and backprojections of multiple cells and voxels respectively, in order to take benefit from the factorizations of the computations induced by the SF approximation. We also present a way to hide data transfers which are known to be the main bottleneck for GPU computing. Next, we validate our new implementation on small volumes, then present results of acceleration on large volumes.

2 Acceleration of the SF pair

2.1 Acceleration of the SF Projector

The SF projection is given by[6]

$$g(u_e, v_e, \phi) = l_{\theta_c}(u_e, v_e) \sum_{x_e, y_e} l_{\phi_v}(\phi; x_e, y_e) F_{trans}(u_e, \phi; x_e, y_e) \sum_{z_{min}}^{z_{max}} F_{ax}(v_e, \phi; x_e, y_e, z_e) f(x_e, y_e, z_e) \quad (1)$$

where $F_{trans}(u_e, \phi; x_e, y_e)$ and $F_{ax}(v_e, \phi; x_e, y_e)$ are transaxial and axial footprints of the voxel, while $l_{\theta_c}(u_e, v_e)$ and $l_{\phi_v}(\phi; x_e, y_e)$ are the amplitude functions given by A2 method.[6] To avoid the conflict between the threads, the GPU implementation of the SF projector is ray-driven.[6]

The original kernel[6] has a lot of intermediate computations which are redundant from one kernel to another, due to the factorizations in formula (1). In order to reduce these redundant computations, the kernel is modified so that it can process several cells per thread in the v-direction : this direction is chosen in order to factorize the computation of the transaxial footprint which requires more time than the axial footprint, since the shape of the transaxial footprint is trapezoidal and the shape of the axial footprint is rectangular in the SF pair.[1] For this purpose, two arrays are created in shared memory in order to avoid to increase the number of registers in the kernel. One array stores axial footprint corresponding to each cell, while the other array stores the product of transaxial and axial footprints. Then the projection is updated.

2.2 Acceleration of the SF Backprojector

For a voxel (x_e, y_e, z_e), the matched SF backprojection is given by[1]

$$b(x_e, y_e, z_e) = \sum_{\phi} l_{\phi_v}(\phi; x_e, y_e) \sum_{v_{min}}^{v_{max}} F_{ax}(v_e, \phi; x_e, y_e, z_e) \times \sum_{u_{min}}^{u_{max}} l_{\theta_c}(u_e, v_e) F_{trans}(u_e, \phi; x_e, y_e) g(u_e, v_e, \phi) \quad (2)$$

The original kernel[6] computes the backprojection of a voxel (x_e, y_e, z_e). The kernel is modified so that one thread can process a column of voxels in the z-direction. At one projection angle ϕ, we store in shared memory all the transaxial footprints of each cell and pick-up for each voxel the transaxial footprints required for the computation.

Then, for each voxel, the sum of the axial and transaxial products is stored in shared memory. The dimension of this array is equal the number of z_e processed by each thread. Algorithm 1 summaries the pseudo-code of our implementation of the SF backprojector.

All the values of the arrays in our modified kernel are stored in the global memory which is a very slow memory. To optimize the access data, the arrays are stored in the shared memory and to guarentee a unique access, the dimension of the arrays are multiplied by the size of the block.

Algorithm 1 Proposed SF backprojector

Initialize $B_{back}[z_{e_0} : z_{e_0} + K - 1] = 0$
for each ϕ **do**
  Compute $v_{min}(z_{e_0})$ and $v_{max}(z_{e_0} + K - 1)$
  Initialize $\tilde{F}_{trans}[v_{min}(z_{e_0}) : v_{min}(z_{e_0} + K - 1)] = 0$
  for $u_e = u_{min}$ to u_{max} **do**
    Compute F_{trans}
    for $v_e = v_{min}(z_e)$ to $v_{max}(z_e + K - 1)$ **do**
      $\tilde{F}_{trans}[v_e] += F_{trans}\ l_{\theta_c}(u_e, v_e) g(u_e, v_e, \phi)$
    end for
  end for
  for $z_e = z_{e_0}$ to $z_{e_0} + K - 1$ **do**
    Compute $v_{min}(z_e)$ and $v_{max}(z_e)$ and initialize $B_\phi = 0$
    for $v_e = v_{min}(z)$ to $v_{max}(z)$ **do**
      Compute and store in shared memory $F_{ax}[v_e]$
      $B_\phi += F_{ax}[v_e] \times F_{trans}[v_e]$
    end for
    Compute and store in shared memory $B_{back}[z_e] += l_{\phi_v} \times B_\phi$
  end for
  for $z_e = z_{e_0}$ to $z_{e_0} + K - 1$ **do**
    $B(x_e, y_e, z_e) = B_{back}[z_e]$
  end for
end for

2.3 Acceleration with multi-streaming and multi-GPU

For large volumes, it is not possible to copy all the data on the GPU. Parts of the data are sent to the GPU one after the other and the results are collected by the CPU. Streams are used to make parallelism tasks in order to mask data transfer, so the GPUs send results and receive data during the computation of one block operator. Multi-GPU computations work following the same principle in order to further accelerate the operators.

3 Results

The new kernels are first tested on a volume of 256^3 voxels, with a detector of 256^2 cells and 256 projections, using a NVIDIA TITAN V graphic card. Sizes of the blocks for the projector and the backprojector are set to $16 \times 16 \times 1$. The original volume is shown in figure 1, and the obtained SF projections and backprojection are shown in figures 2 and 3. Table 1 shows the impact on each operator's runtime of the number of cells/voxels handled by each thread. We observe a three fold acceleration for the SF backprojector and a two fold acceleration for the SF projector. The coupling degree[2] of our GPU SF pair is equal to 1.0001 which is very close to 1, which means our SF pair is very well matched,[2] furthermore the coupling degree of our SF pair is not impacted the number of voxels/cells handled by each thread. Table 2 presents the runtime of the new backprojector and projector using multi-GPU and multi-streams on large volume of 1024^3 and 2048^3 voxels. We use

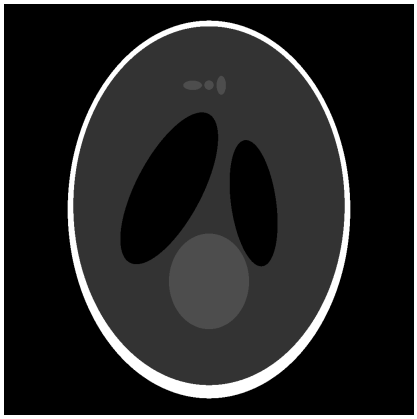

Fig 1 Original volume

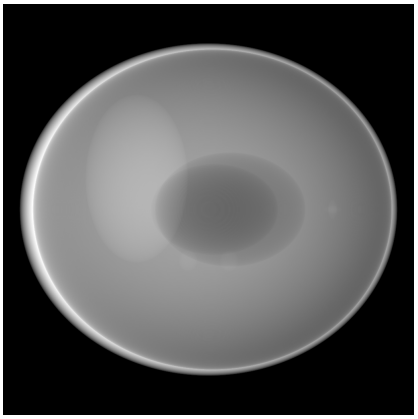

Fig 2 Obtained SF projection

Fig 3 Obtained SF backprojection

Number of Cells	Time Backprojector (ms)	Number of voxels	Time projector (ms)
1	367	1	999
2	221	2	605
4	159	4	**416**
8	**109**	8	438
16	147	16	419

Table 1 Runtime of the projection and backprojection operator depending on the number of cells or voxels.

TITAN X Maxwell GPU, each thread of our backprojector treats 8 voxels and the projector treats 16 cells, the size of the projector block is $128 \times 1 \times 1$ which is fully optimized and the backprojector block is $16 \times 16 \times 1$.

In order to go further in the analysis, we compare the efficiency of our implementation of the matched SF pair with an unmatched pair. The unmatched pair has a Joseph's projector[7] (J) and a voxel-driven (VD) backprojector[8] and is called the J/VD pair in the following. Table 3 compares the SF pair and the J/VD pair on a phantom of size 1024^3 voxels, using 1024 projections of size 1536×1152 pixels. The used GPU is a NVIDIA Titan V. In table 3, we consider the number of Tera-samples in the volume and in the projections respectively read by the projector and the backprojector. For the projector, this number is given by the number of rays multiplied by the number of samples read for each ray, while, for the backprojector, it corresponds to the number of voxels multiplied by the number of projections multiplied by the number of samples on the detector read for each voxel. We see that the SF pair requires to read more samples than the J/VD pair due its higher accuracy. This explains its higher computation time. Table 3 also shows the number of floating-point operations (FLOP) performed by each pair. We see that this number is much higher for the SF pair than for the J/VD pair. This is due to the fact that the unmatched J/VD pair takes benefit from the fact that its required interpolations are performed by the hardware of the GPU thanks to the use of the texture memory, while there is no obvious way to take benefit from this mechanism for the SF pair.

4 Conclusion

Multi-cell, multi-voxel and multi-GPU computations accelerate the runtime of SF forward and backward projectors. The major factor of acceleration is the number of GPUs followed by the number of

Operator	1 Stream,1 GPU	4 Stream 1 GPU	1 Streams 8 GPUs	4 Streams 8 GPUs
P SF 1024^3 voxels	255686 ms	255256 ms	39084 ms	38126 ms
BP SF 1024^3 voxels	95787 ms	93490 ms	18336 ms	17929 ms
P SF 2048^3 voxels	4260676 ms	4254497 ms	599496 ms	584414 ms
BP SF 2048^3 voxels	1530856 ms	1522824 ms	244798 ms	233101 ms

Table 2 Runtime of the projection (P) and backprojection (BP) operator depending on the number of streams and GPUs

Operator	Tera-samples	Tera-FLOP	FLOP/sample	Time (s)	Tera-samples/s
P J[7]	3.96	3.47	0.88	5.14	0.77
P SF	6.22	102.74	16.5	61.9	0.10
BP VD[8]	4.40	1.72	0.39	1.37	3.23
BP SF	9.74	145.2	14.9	33.8	0.29

Table 3 Efficiency of the matched SF pair *versus* unmatched J/VD pair, on a volume of 1024^3 voxels with 1024 projections of size 1536×1152 pixels

voxels and cells handled by each thread. For small volumes, one GPU can be used. The number of streams provide marginal acceleration gains since the performances of the SF pair are bounded by the computations. Furthermore, by measuring the number of samples read by each operator, as the number of floating-point operations, we have emphasized that the SF pair requires more computations than an unmatched pair, leading to a higher computation time but also to a higher accuracy.

References

1 Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for X-ray CT using separable footprints," *IEEE transactions on medical imaging* **29**(11), 1839–1850 (2010).

2 F. Arcadu, M. Stampanoni, and F. Marone, "On the crucial impact of the coupling projector-backprojector in iterative tomographic reconstruction," *arXiv preprint arXiv:1612.05515* (2016).

3 B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Physics in medicine and biology* **49**(11), 2463 (2004).

4 M. Wu and J. A. Fessler, "GPU acceleration of 3D forward and backward projection using separable footprints for X-ray CT image reconstruction," in *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, **6**, 021911, Citeseer (2011).

5 X. Xie, M. G. McGaffin, Y. Long, *et al.*, "Accelerating separable footprint (SF) forward and back projection on GPU," in *SPIE Medical Imaging*, 101322S–101322S, International Society for Optics and Photonics (2017).

6 C. Chapdelaine, N. Gac, A. Mohammad-Djafari, *et al.*, "New GPU implementation of Separable Footprint Projector and Backprojector : first results," in *The 5th International Conference on Image Formation in X-Ray Computed Tomography*, 314–317 (2018).

7 P. M. Joseph, "An improved algorithm for reprojecting rays through pixel images," *IEEE transactions on medical imaging* **1**(3), 192–196 (1982).

8 N. Gac, S. Mancini, M. Desvignes, *et al.*, "High speed 3D tomography on CPU, GPU, and FPGA," *EURASIP Journal on Embedded systems* **2008**, 5 (2008).